配置 Babel（进阶）

大多数人使用 Babel 的内建预设就足够了，不过 Babel 提供了更多更细粒度的能力。

手动指定插件

Babel 预设就是一些预先配置好的插件的集合，如果你想要做一些不一样的事情你会手动去设定插件，这和使用预设几乎完全相同。

首先安装插件：

```
$ npm install --save-dev babel-plugin-transform-es2015-classes
```

然后往 `.babelrc` 文件添加 `plugins` 字段。.

```
  {
+   "plugins": [
+     "transform-es2015-classes"
+   ]
  }
```

这能让你对正在使用的转换器进行更细致的控制。

完整的官方插件列表请见 Babel 插件页面。.

同时也别忘了看看由社区构建的其他插件。 如果你想学习如何编写自己的插件可以阅读 Babel 插件手册。.

插件选项

很多插件也有选项用于配置他们自身的行为。 例如，很多转换器都有“宽松”模式，通过放弃一些标准中的行为来生成更简化且性能更好的代码。

要为插件添加选项，只需要做出以下更改：

```
  {
    "plugins": [
-     "transform-es2015-classes"
+     ["transform-es2015-classes", { "loose": true }]
    ]
  }
```

基于环境自定义 Babel

巴贝尔插件解决许多不同的问题。 其中大多数是开发工具，可以帮助你调试代码或是与工具集成。 也有大量的插件用于在生产环境中优化你的代码。

因此，想要基于环境来配置 Babel 是很常见的。你可以轻松的使用 `.babelrc` 文件来达成目的。

```
 {
   "presets": ["es2015"],
   "plugins": [],
+  "env": {
+    "development": {
+      "plugins": [...]
+    },
+    "production": {
+      "plugins": [...]
+    }
   }
 }
```

Babel 将根据当前环境来开启 `env` 下的配置。

当前环境可以使用 `process.env.BABEL_ENV` 来获得。 如果 `BABEL_ENV` 不可用，将会替换成 `NODE_ENV`，并且如果后者也没有设置，那么缺省值是`"development"`。.

Unix

```
$ BABEL_ENV=production [COMMAND]
$ NODE_ENV=production [COMMAND]
```

Windows

```
$ SET BABEL_ENV=production
$ [COMMAND]
```

注意：`[COMMAND]` 指的是任意一个用来运行 Babel 的命令（如：`babel`，`babel-node`，或是 `node`，如果你使用了 register 钩子的话）。

提示：如果你想要让命令能够跨 unix 和 windows 平台运行的话，可以使用 `cross-env`。